WEST

Generate Collection | Print

L7: Entry 2 of 5 File: USPT May 27, 2003

DOCUMENT-IDENTIFIER: US 6571285 B1
TITLE: Providing an integrated service assurance environment for a network

Abstract Text (1):
A method providing service assurance for a network to maintain an agreed upon Quality of Service. First, an alarm is generated to indicate a status of a network. The generation of the alarm comprises selecting a parameter of network to be monitored; determining a triggering level of the parameter; monitoring the parameter of an occurrence of the triggering level; and initiating alarm notification upon the monitored occurrence of the triggering level. Network event information is then dispatched upon generation of the alarm and is subsequently mapped. The data collected on the status of the network is then manipulated by concatenating the data collected on a network into a master file; reformatting the data into a standarized format; translating the data to key codes; sorting the data according to predetermined criteria; and concatenating the sorted data together. The data is then sorted in a database. Thereafter, network availability is conveyed graphically.

Brief Summary Text (2):
The present invention relates to service assurance environments and more particularly to an integrated service assurance environment for a network.

Brief Summary Text (8):
Despite the foregoing efforts, network failures are inevitable, and there is a need of monitoring network performance for the purpose of maintaining a predetermined agreed upon QoS.

Brief Summary Text (10):
A method providing service assurance for a network to maintain an agreed upon Quality of Service. First, an alarm is generated to indicate a status of a network. The generation of the alarm comprises selecting a parameter of network to be monitored; determining a triggering level of the parameter; monitoring the parameter of an occurence of the triggering level; and initiating alarm notification upon the monitored occurrence of the triggered level. Network event information is then dispatched upon generation of the alarm and is subsequently mapped. The data collected on the status of the network is then manipulated by concatenating the data collected on a network into a master file; reformatting the data into a standarized format; translating the data to key codes; sorting the data according to predetermined criteria; and concatenating the sorted data together. The data is then stored in a database. Thereafter, network availability is conveyed graphically.

Drawing Description Text (4):
FIG. 2 illustrates an embodiment of the present invention which provides service assurance for a network;

Drawing Description Text (5):
FIG. 3 illustrates one embodiment of the present invention for dispatching network event information of a network with service assurance capabilities;

Drawing Description Text (9):
FIG. 7 illustrates an embodiment of the present invention for retrieving and relocating event and performance data of a network with service assurance capabilities;

Drawing Description Text (12):
FIG. 10 depicts an embodiment of the present invention which generates an alarm to indicate a status of a network for service assurance purposes;

Drawing Description Text (18):
FIG. 16 depicts an embodiment of the present invention which graphically conveys availability in a network with service assurance capabilities

Drawing Description Text (34):
FIG. 31 illustrates an embodiment of the present invention which maps events on a network with service assurance capabilities; and

Detailed Description Text (30):
One embodiment of the present invention is composed of multiple software programs which are linked together to create an architecture which is capable of monitoring a network for events and checking system functions and resources. Such events can include alarms, faults, alerts, etc. Other embodiments of the present invention may each include an individual software program.

Detailed Description Text (32):
Accordingly, FIG. 2 illustrates an embodiment of the present invention which provides service assurance for a network. In operation 200, an alarm is generated to indicate a status of a network. Network event information of the network is dispatched in operation 202 upon generation of the alarm after which the network event information is mapped in operation 204. The data collected on the status of the network is manipulated and stored in a database. See operations 206 and 208. In operation 210, availability of the network is conveyed graphically.

Detailed Description Text (38):
Accordingly, FIG. 3 illustrates one embodiment of the present invention for dispatching network event information of a network with service assurance capabilities. In operation 300, a network is monitored for an event. Thereafter, in operation 302, at least one notification action is generated based upon the occurrence of the event. The notification action may include an alphanumeric page, an e-mail message, a resolution script, a remedy trouble ticket, and/or a log message. Further, the notification action may be transmitted in operation 304 to notify a recipient about the occurrence of the event.

Detailed Description Text (54):
In yet another aspect of the present invention, the received data relates to a parameter of a network. Further, the data may be stored for service assurance purposes on the network.

Detailed Description Text (64):
One embodiment of the present invention provides the ability to correlate network events to individual customers (or providers in a Managed Network Services world) and notify customer service representatives of known outages affecting customers through a web interface. This allows proactive notification to customers of problems that affect them as well as builds confidence in customers calling to report problems that the provider is aware of.

Detailed Description Text (67):
Referring to FIG. 5, in one embodiment of the present invention, an activation signal is received in operation 500. Upon receipt of the activation signal, a signal is transmitted in operation 502 to initiate the retrieving of network performance data and network event data generated from at least one network monitor. Such network monitor is adapted for monitoring a network system and the relocating of the data into a common directory. Then, in operation 504, the signal is transmitted to initiate the manipulation of the data and the loading of the manipulated data into a database.

Detailed Description Text (71):
The following subsections describe an embodiment of the present invention that controls the collection, manipulation and storage of network performance data and network event data of a network with service assurance capabilities and provides an exemplary step-by-step overview of the flow of data from collection to when it's loaded into the database. FIG. 6 is a flowchart that provides an overview of a data collection process of one embodiment of the present invention.

Detailed Description Text (74):
The data collection is started by the network monitory applications creating their ASCII text data files. These files are generally stored locally on the machines they are running on. Specifics on where these files should be stored are located in the installation & configuration instructions for each application.

Detailed Description Text (129):
The information contained in each entry is unique to the data being retrieved. The ".about." character is used as field delimiter throughout the file. Following is a description of the fields that must be defined for each entry: $NodeName--DNS name or alias for the local/remote host from which files need to be collected. $Location--Options include "local" or "remote". $SourcePath--Complete source path designating the directory from which data will be retrieved. $SourceExt--File extension used to designate which files need to be collected. $TargetPath--Complete target path designating the destination directory, on $DestHost, where files should be transferred to. $TmpExt $MoveMethod $Archive $ArchivePath $ArchiveExt $TransferType $Platform $TargetExt $DestHost--DNS name or alias for the host where files will to be transferred to. It is not necessary for the destination host to be the system calling get_data.pl. $LogFile $UnixRemoteScript

Detailed Description Text (132):
FIG. 7 illustrates an embodiment of the present invention for retrieving and relocating event and performance data of a network with service assurance capabilities. First, in operation 700, a data file is obtained from a host. The data file includes event data collected on a network and/or performance data collected on the network. In operation 702, a verification control file is created that is associated with the data file. The data file is renamed in operation 704 and copied to a target directory in operation 706. Thereafter, the copying of the renamed data file is verified with the verification control file in operation 708.

Detailed Description Text (223):
Accordingly, FIG. 10 depicts an embodiment of the present invention which generates an alarm to indicate a status of a network for service assurance purposes. Such purposes can include identifying errors and faults, monitoring system resources, anticipating problems, etc. Once a parameter of a network that is to be monitored is selected in operation 1000, a triggering level of the parameter is determined in operation 1002. In operation 1004, the parameter for an occurrence of the triggering level is monitored. If the triggering level is reached, an alarm is initiated in operation 1006.

Detailed Description Text (257):
The database for the Service Assurance Toolkit can be designed as a data warehouse. This design offers greater performance and flexibility, which will enable the database to evolve in future releases of the Service Assurance Toolkit. The architecture for a successful data warehouse, by which we mean the end-to-end tools and processes, will vary from implementation to implementation. A typical data warehousing architecture should include, as a minimum: Multiple extract programs from one or more operational systems, to retrieve the source data for the warehouse. A data repository ("Operational Data Store") containing the extracted data in an appropriate model. A tool to analyze and display the data as reports and charts.

Detailed Description Text (509):
FIG. 16 depicts an embodiment of the present invention which graphically conveys availability in a network with service assurance capabilities. In operation 1600, report parameters are selected relating to availability of monitored elements, services, and processes of a network. A database is polled in operation 1602 for data that matches the report parameters. A graph is generated in operation 1604 from the data that matches the report parameters. In operation 1606, the generated graph is displayed to graphically represent the monitored elements, services, and processes of the network.

Detailed Description Text (553):
FIGS. 19-22 provide historical record of collected performance data and network events. Exception reporting is limited to views of events that occurred in real-time and does not include finding exceptions in the historical data by analyzing past data.

Detailed Description Text (680):
Web Site Tab- Change description name as desired (Service Assurance Web Site). Change IP address to your own (149.122.57.21--it also happens to be in the drop down box) Check `enable logging` and under properties change the log file directory (C:.backslash.data.backslash.iislogs)

Detailed Description Text (684):
FTP Site Tab- Check `enable logging` and under properties change the log file directory (C:.backslash.data.backslash.iislogs) Change description name as desired (Service

Assurance FTP Site). Right click on Administration Web Site and under Properties change these settings:

Detailed Description Text (685):
Web Site Tab- Check `enable logging` and under properties change the log file directory (C:.backslash.data.backslash.iislogs) Change description name as desired (Service Assurance Administration Web Site). Add a `cgi-bin` folder into the directory where your http pages are located

Detailed Description Text (697):
SPSS Database Capture Wizard In SPSS Data Editor go to Edit>Database Capture>New Query to open the ODBC Wizard. Choose data source to retrieve data and click Next. Drag table onto right box to see fields and click Next. (Can also select a subset of fields here) Limit Retrieved Cases page, click Next. (Can select a subset of cases based on conditional expressions in this dialog box) Define Variables page, click Next. (Can specify more user friendly variable names here) Results page. Change column names into SPSS Syntax to desired names. The names to use are located by opening SQL+ and on the command line typing the table name. i.e.>describe spss_test

Detailed Description Text (767):
The working data file is the data file you build to use in the current session. You can retrieve an SPSS-format data file using GET, which in effect makes a working copy of the specified file. The working data file is not created until SPSS encounters a command (usually a procedure) that causes it to read the data. At this point, SPSS executes all of the preceding data definition and transformation commands and the command that causes the data to be read. The working data file is then available for further transformations and procedures, and it remains available until replaced by a new working data file or until the end of the session.

Detailed Description Text (770):
(***This includes the GET CAPTURE command that retrieves data from a database and converts them to a format that can be used by program procedures. GET CAPTURE retrieves data and data information and builds a working data file for the current session.)

Detailed Description Text (872):
Introduction This section is intended as a lessons-learned encapsulation for ECM environment configurations. This section is intended as an aid in the creation and modification of event correlation models/alarm definitions in ECM. Included in this section are references to event correlation models/alarm definitions and PERL scripts that were devised in the development of this component of the Service Assurance final phase 2 deliverable. These pieces are included in the phase 2 repository for importation into an install of ECM.

Detailed Description Text (926):
Once the actual trap has been received into the system, the parsing of the trap has to be done through PERL in most cases. There are other means to deal with traps, but PERL is native to ECM. This led to its' use in the phase two development of this product for the Service Assurance project.

Detailed Description Text (928):
Utilizing these "Vb" variables, you can get at the variable bindings that are appended to the end of the SNMP trap that was transmitted via PATROLLER. As this is where the actual data resides, these variables make extraction and correlation a breeze. PATROLLER builds its' variable bindings in a single dimension, and to retrieve this data requires nothing more than extraction of the `VbValue(0)` variable. This comes in as straight text in the form of "Application:Instance:Parameter:State:Value".

Detailed Description Text (961):
FIG. 31 illustrates an embodiment of the present invention which maps events on a network with service assurance capabilities. In operation 3100, a network is monitored for the occurrence of availability events, threshold events, and trap events. At least one occurred event is correlated to at least one other occurred event in operation 3102 to generate at least one correlating event. In operation 3104, the occurred events and correlating events are mapped on at least one network map. The network map is subsequently displayed in operation 3106.

Detailed Description Text (962):
In one embodiment of the present invention, the step of monitoring the network further comprises: tracking the availability of individual components of network for events,

tracking the availability of individual services of the network for events, tracking the availability of individual processes of an operating system of the network for events, tracking the status of agent processes on individual components of the network for events, monitoring the operating system and application performance of network for threshold events, and monitoring traps of the network for events.

Detailed Description Text (963):
In yet another embodiment, the network map is a node level map and/or an event level map. The node level map displays node responding events, agent not responding events, and/or node down events. The step of mapping the occurred events and correlating events when the network map comprises the event level map further comprise of additional steps. In particular, the occurred events and correlating events may be filtered based upon predetermined criteria. The filtered events may also be mapped on the event level map. In still yet another embodiment, at least one notification action is generated based upon the occurred events and/or correlating events.

Detailed Description Text (1016):
This section will discuss the Service Assurance test environment specific details of the Collector Internet Service Monitors (ISM's) and requirements for a remote installation.

Detailed Description Text (1069):
This section is defined to allow for a single point of definition for event codes within the Service Assurance project.

Detailed Description Text (1110):
Requirements for the Development Environment The ability to check in and check out files so that only one person is editing a file at a time The following will be tracked on each file that is being version controlled. date of creation, date of last modification, version, and change history (annotated with date, rev, user, and comments). The ability to associate revision numbers with development environment (the environment can be set to `dev`, `tst`, or `prd`). This make is it make it possible to develop multiple releases at one time. The ability to retrieve previous versions of a component or sub-component for either edit or review. Support for multiple development languages. Ability for users to operate in a separate environment. This includes operations on the users `own` test data and executables. Backup and recovery of source code, documentation, test data, etc... Tools to aid in the debugging of components and sub-components. This would include generation of test data and unit test conditions. Documentation for users on how to use the tools under different conditions and situations. Ability to tie SIR or defect number to all components and sub-components that are. Ability to migrate components and sub-components between environments. Documented coding standards for each type of development language used). Provide shells as a starting point for each coding language used. Strategy for software distribution.

Detailed Description Text (1114):
Installation of an exemplary Service Assurance Toolkit can be broken down into two parts. 1. Installation and configuration of software. Network Node Manager Event Correlator and Manager Collector Patroller Database Software Telalert Server Software SPSS Internet Information Server 2. Installation and customization of Service Assurance Glueware. Determination of an appropriate directory structure. Customization of all environment specific settings in the Glueware scripts. This includes variables that are local to each script as well as global variables from the SACommon.pm Perl Module. Also, care should be taken regarding hard coded environment specific information in each script.

Detailed Description Text (1137):
This section is intended to list the hardware inventory, software instalation locations, and software requirements of the Service Assurance development test network. This section will summarize the detailed findings of an Excel workbook that is accessible in: Functional Repository.fwdarw.Capability Analysis.fwdarw.Hardware/Software Expense

Detailed Description Text (1140):
This section will list Service Assurance's responsibilities to an exemplary network to insure timely backup and recovery.

Detailed Description Text (1143):
An individual from the project should be identified to receive backup completion notices. These notices are mailed at the completion of each nightly backup cycle. This

person should then verify that all Service Assurance servers were adequately backed up the previous night.

Detailed Description Text (1145):
This section provides an overview of exemplary steps to build the Service Assurance environment. The procedures are presented as an ordered list, and unless denoted by `*`, should be performed in their respective order. Procedures denoted by `*` are independent, and can be performed out of order with respect to other procedures at their respective level in the hierarchy.

Detailed Description Text (1160):
The base requirements to run all of the Service Assurance applications concurrently on one Sun Solaris system are: Sun Ultra 2 server Dual 200 MHz (or faster) CPUs 768 MB RAM 2-4GB Internal SCSI disks (mirrored) for system .about.20 GB external disk storage (mirrored and striped)

Detailed Description Text (1164):
As discussed in the requirements section, dual (or more) CPUs and a large amount of physical RAM are crucial to the performance of the system. Mirroring of the system disk is critical for maintaining availability of the Service Assurance system. Mirroring and striping is crucial on the external drives to provide the performance and throughput required by the real-time data gathering portions of the system.

Detailed Description Text (1172):
Details of using the Version Control System (VCS) The VCS should only be used on files that use `#` to denote comments. This is due to the vcs header being framed with `#`. Ownership of the files will be noc:twsa, twsa is a nis group to which members of the Service Assurance team belongs. The ability to check in and check out files so that only one person is editing a file at a time. A lock file is created in the /sa/vcs/source which controls usage. When a file is checked in; its file permissions are 544. When a file is checked out; its file permissions are 744. Each time a file is checked out, edited and checked back in; the revision of that file is incremented. For Example, when version 1.9 is checked back in it becomes version 1.10.

Detailed Description Text (1177):
This is the primary command for checking a file out (for editing purposes) of the vcs. A lock on the specified file is created and file permissions are set to allow the user to edit the file. The user must manually call vi (or text editor of choice) to edit the file. This command can also be used to retrieve a previous version of a file.

Detailed Description Text (1181):
Retrieves a read-only copy of a file for viewing purposes. No lock file is created. This command can be used to get a copy offile for off-line editing in a user's home directory. This command can also be used to retrieve a previous version of a file or recover when a file has been accidently deleted.

Detailed Description Paragraph Table (21):
SQL*Plus Script File set echo off set heading off set embedded on set pagesize 1000 set termout off spool &1 select name from v$datafile; select member from v$logfile; select name from v$controlfile; spool off set termout on set heading on set feedback on set embedded off exit Password Script #!/bin/sh #--# # Script_name: getpass # Description: This script will be used to retrieve the # # appropriate password from the .password # # file. It can be used from the command # # line to retrieve a password or from a # # shell script to eliminate hard coding of # # passwords. The .password file is located # # in $DBSE_HOME, with the executable # # located in $DBSE_HOME/bin. # # # # Dependencies: getpass requires one file, .password # # # # SID_FILE: Contains a list passwords for # # system, sys, and dbse. # # # # Command syntax: getpass USERNAME # #--# # what shell do we use? SHELL=/usr/bin/sh # where is our home? DBSE_HOME=/files2/ipsa/vendor/dbse/product/7.3.4; export DBSE.sub.-- HOME # what path do we look for? PATH=/usr/bin:/bin:${DBSE_HOME}/bin; export PATH # who are we retrieving the password for if[$1] then USER=$1 else echo"" echo "USAGE: getpass 'username'" echo"" exit 1 fi # make sure the .password file exist if[! -s $DBSE_HOME/.password] then echo"" echo "ERROR: this machine does not appear to have a password file" echo"" exit 1 fi # get the appropriate password PASSWORD='cat $DBSE_HOME/.password.vertline.grep -i "${USER} ".vertline.awk' {print $2}" BAD_SIDS='cat $DBSE_HOME/.password.vertline.grep -i "${USER} ".vertline.awk' {print $3}" OLD_PASS='cat $DBSE_HOME/.password.vertline.grep -i "${USER} ".vertline.awk'

{print $4}" # if a password was found, print it to the screen if[$PASSWORD] then echo "${PASSWORD}" # print a message if passwords appear to not be synced between databases if[$BAD_SIDS] then BAD_SIDS='echo ${BAD_SIDS}.vertline.cut-c2-100' echo"" echo "WARNING:.backslash.tThe password for ${USER} may not be synchronized" echo ".backslash.t.backslash.tbetween all databases. The database(s) ${BAD_SIDS} appear(s)" echo ".backslash.t.backslash.tto use the old password '${OLD_PASS}'" echo"" fi else echo"" echo"ERROR: ${USER} is not a supported username" echo"" exit 1 fi

Detailed Description Paragraph Table (23):
Password Script #!/bin/sh
#--# # Script_name: getpass # # Description: This script will be used to retrieve the # # appropriate password from the .password # # file. It can be used from the command # # line to retrieve a password or from a # # shell script to eliminate hard coding of # # passwords. The .password file is located # # in $DBSE_HOME, with the executable # # located in $DBSE_HOME/bin. # # # # Dependencies: getpass requires one file, .password # # # # SID_FILE: Contains a list passwords for # # system, sys, and dbse. # # # # Command syntax: getpass USERNAME # #--# # what shell do we use? SHELL=/usr/bin/sh # where is our home? DBSE_HOME=/files2/ipsa/vendor/dbs/product/7.3.4; export DBSE.sub.-- HOME # what path do we look for? PATH=/usr/bin:/bin:${DBSE_HOME}/bin; export PATH # who are we retrieving the password for if[$1] then USER=$1 else echo"" echo "USAGE: getpass 'username'" echo"" exit 1 fi # make sure the .password file exist if[!-s $DBSE_HOME/.password] then echo"" echo "ERROR: this machine does not appear to have a password file" echo"" exit 1 fi # get the appropriate password PASSWORD='cat $DBSE_HOME/.password.vertline.grep -i "${USER} ".vertline.awk ' {print $2}" BAD_SIDS='cat $DBSE_HOME/.password.vertline.grep -i "${USER} ".vertline.awk ' {print $3}" OLD_PASS='cat $DBSE_HOME/.password.vertline.grep -i "${USER} ".vertline.awk ' {print $4}" # if a password was found, print it to the screen if[$PASSWORD] then echo"${PASSWORD}" # print a message if passwords appear to not be synced between databases if[$BAD_SIDS] then BAD_SIDS='echo ${BAD_SIDS}cut -c2-100' echo "" echo "WARNING:.backslash.tThe password for ${USER} may not be synchronized echo ".backslash.t.backslash.tbetween all databases. The database(s) ${BAD_SIDS } appear(s)" echo ".backslash.t.backslash.tto use the old password `${OLD_PASS}'" echo "" fi else echo "" echo "ERROR: ${USER} is not a supported username" echo "" exit 1 fi

Detailed Description Paragraph Table (27):
getpass (Born Shell) #!/bin/sh
#--# # Script_name: get_pass # # # # Description: This script will be used to retrieve the # # appropriate password from the password # # file. It can be used from the command # # line to retrieve a password or from a # # shell script to eliminate hard coding of # # passwords. The password file is located # # in $DBSE_HOME, with the executable # # located in $DBSE.sub.' HOME/bin. # # # # Dependencies: getpass requires one file, .password # # # # SID_FILE: Contains a list passwords for # # system, sys, and dbse. # # # # Command syntax: getpass USERNAME # #--# #what shell do we use? SHELL=/usr/bin/sh #where is our home? DBSE_HOME=/files0/ipsa/vendor/dbse/product/7.3.4; export DBSE_HOME #what path do we look for? PATH=/usr/bin:/bin:${DBSE_HOME}/bin; export PATH #who are we retrieving the password for if[$1] then USER=$1 else echo " " echo "USAGE: getpass `username`" echo " " exit 1 fi #make sure the .password file exist if [! -s $DBSE_HOME/.password] then echo " " echo "ERROR: this machine does not appear to have a password file" echo " " exit 1 fi #get the appropriate password PASSWORD=`cat $DBSE_HOME/.password.vertline.grep -i "${USER} ".vertline.awk`{print $2}" BAD_SIDS=`cat $DBSE_HOME/.password.vertline.grep -i "${USER} ".vertline.awk`{print $3}" OLD_PASS=`cat $DBSE_HOME/.password.vertline.grep -i "${USER} ".vertline.awk`{print $4}" #if a password was found, print it to the screen if[$PASSWORD] then echo "${PASSWORD}" #print a message if passwords appear to not be synced between databases if[$BAD_SIDS] then BAD_SIDS=`echo ${BAD_SIDS}.vertline.cut -c2-100` echo " " echo "WARNINC:.backslash.tThe password for ${USER} may not be synchronized" echo ".backslash.t.backslash.tbetween all databases. The database(s) ${BAD_SIDS} appear(s)" echo ".backslash.t.backslash.tto use the old password `${OLD_PASS}`" echo " " fi else echo " " echo "ERROR: ${USER} is not a supported username" echo " " exit 1 fi get_index (PL/SQL Script) SET ECHO OFF REM Procedure Name: get_index REM REM Description: Procedure creates a file with a list of current indexes VARIABLE T_Name varchar2(30) SET SERVEROUTPUT OFF SET VERIFY OFF SET FEEDBACK OFF SET FLUSH OFF SET TRIMSPOOL ON SET TERMOUT OFF ACCEPT T_Name char SET PAGES 0 SET HEADING OFF SPOOL /files6/ipsa/data_loads/&T_Name..idx select index_name from dba_indexes where table_name = UPPER(`&T_Name`); SPOOL OFF SET TERMOUT ON SET SERVEROUTPUT OFF SET VERIFY

ON SET FEEDBACK ON SET ECHO ON drop_index_p (PL/SQL Stored Procedure) CREATE OR REPLACE PROCEDURE drop_index_p (IN_IndexName IN VARCHAR2) AS ws_stmt varchar2(100); ws_owner varchar2(30); CURSOR c_index_owner (IN_IndexName VARCHAR2) IS SELECT table_owner FROM dba_indexes WHERE index_name = UPPER(IN_IndexName); BEGIN OPEN c_index_owner(IN_IndexName); FETCH c_index_owner INTO ws_owner; ws_stmt := `DROP INDEX `.vertline..vertline.ws_owner.vertline..vertline.`. `.vertline..vertline.IN_IndexName; p_exec (ws_stmt); CLOSE c_index_owner; END drop_index_p; / show errors; create_ctl (PL/SQL Script) SET ECHO OFF REM Procedure Name: create_ctl REM Description: Procedure created control file required by SQL*Loader REM Uses create_ctl_.sql to generate control file REM Tables Accessed: All_Tab_Columns (Data Dictionary Table) VARIABLE T_Name varchar2(30) SET SERVEROUTPUT OFF SET VERIFY OFF SET FEEDBACK OFF SET TRIMSPOOL ON SET TERMOUT OFF SET FLUSH OFF ACCEPT T_Name char delete control_tb; commit; EXECUTE CREATE_CTL_P(`&T_NAME`); SET PAGES 0 SET HEADING OFF SPOOL /files6/ipsa/data_loads/control_files/&T_Name..ctl select line_text from control_tb order by line_nbr; SPOOL OFF SET TERMOUT ON SET SERVEROUTPUT OFF SET VERIFY ON SET FEEDBACK ON SET ECHO ON create_ctl_p, (PL/SQL Stored Procedure) **Note-there should be a table used specifically for this procedure (control_tb) with the columns LINE_NBR number(3) and LINE_TEXT varchar2(100). CREATE OR REPLACE PROCEDURE create_ctl_p (IN_TableName IN user_tab_columns.table_name%TYPE) AS WS_RecNo number(2) := 0; WS_line_ctr number(3) := 0; WS_index_name varchar2(30); CURSOR c_index_name(IN_TableName IN user_tab_columns.table_name%TYPE) is SELECT index_name FROM sys.dba_indexes WHERE sys.dba_indexes.table_name = UPPER(IN_TableName) AND sys.dba_indexes.index_name like `PK%`; CURSOR c_user_tab_columns(IN_TableName user_tab_columns.table_name%type) is SELECT column_name, data_type, data_length FROM user_tab_columns WHERE user_tab_columns.table_name UPPER(IN_TableName); BEGIN OPEN c_index_name(IN_TableName); FETCH c_index_name INTO WS_index_name; insert into control_tb values (WS_line_ctr,`UNRECOVERABLE`); WS_line_ctr := WS_line_ctr + 1; insert into control_tb values (WS_line_ctr,`LOAD DATA`); WS_line_ctr := WS_line_ctr + 1; insert into control_tb values (WS_line_ctr,`INFILE "/files6/ipsa/data_loads/data_files/ `.vertline..vertline.IN_TableName.vertline..vertline.`.dat`"); WS_line_ctr := WS_line_ctr + 1; insert into control_tb values (WS_line_ctr,`APPEND`); WS_line_ctr := WS_line_ctr + 1; insert into control_tb values (WS_line_ctr, `INTO TABLE`.vertline..vertline.IN_TableName); WS_line_ctr := WS_line_ctr + 1; insert into control_tb values (WS_line_ctr,`SORTED INDEXES (`.vertline..vertline.WS_index_name.vertline..vertline.`)`); WS_line_ctr := WS_line_ctr + 1; insert into control_tb values (WS_line_ctr,`FIELDS TERMINATED BY `.vertline..vertline.`","`); WS_line_ctr := WS_line_ctr + 1; insert into control_tb values (WS_line_ctr,`(`); WS_line_ctr := WS_line_ctr + 1; FOR c_rec in c_user_tab_columns(IN_TableName) LOOP IF WS_RecNo = 0 then IF c_rec.data_type = `DATE` THEN insert into control_tb values (WS_line_ctr,c_rec.column_name.vertline..vertline. `DATE "MM/DD/YYYY HH24:MI:SS`"); ELSE insert into control_tb values (WS_line_ctr,c_rec.column_name); END IF; WS_RecNo := 1; ELSE IF c_rec.data_type = `DATE` THEN insert into control_tb values (WS_line_ctr,`,`.vertline..vertline.c_rec.column_name.vertline..vertline. `DATE "MM/DD/YYYY HH24:MI:SS`"); ELSE insert into control_tb values (WS.sub.-line_ctr,`,`.vertline..vertline.c_rec.column_name); END IF; END IF; WS_line_ctr := WS_line_ctr + 1; END LOOP; insert into control_tb values (WS_line_ctr,`)`); EXCEPTION WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE(`The table does not exist:`); END; / show errors p_exec (PL/SQL Stored Procedure) CREATE OR REPLACE PROCEDURE p_exec (IN_String IN varchar2) AS c_cursor INTEGER; WS_ret INTEGER; BEGIN c_cursor := DBMS_SQL.OPEN_CURSOR; DBMS_SQL.PARSE(c_cursor IN_String, DBMS_SQL.V7); WS_ret := DBMS_SQL.EXECUTE(c_cursor); DBMS_SQL.CLOSE_CURSOR(c_cursor); END; / sp_kill_users_p (PL/SQL Stored Procedure) CREATE OR REPLACE PROCEDURE sp_kill_users_01 AS ws_sid number(22); ws_serial# number(22); ws_stmt varchar2(100); ora_err_code BINARY_INTEGER;

Detailed Description Paragraph Table (32):
getpass (Born Shell) #!/bin/sh #----------------------# #Scipt_name: get_pass # # # #Description: This script will be used to retrieve the # #appropriate password from the password # #file. It can be used from the command # #line to retrieve a password or from a # #shell script to eliminate hard coding of # #passwords. The .password file is located # #in $DBSE_HOME, with the executable # #located in $DBSE_HOME/bin # #Dependencies: get pass requires one file, password # #SID_FILE: Contains a list of passwords for # #system, sys, and dbse. # #Command syntax: getpass USERNAME # #------------------# #what shell do we use 48 SHELL = /usr/bin/sh #where is our home? DBSE_HOME =/files0/ipsa/vendor/dbse/product/ 7.3.4; export DBSE_HOME #what path do we look for? PATH=/usr/bin:/bin:${DBSE_HOME}/bin; export PATH #who are we retrieving the password for if[$1] then USER =$1 else echo "" echo "USAGE: getpass 'username'" echo"" exit 1 fi #make sure the password file exist if[!-s $DBSE_HOME/.password] then echo echo "ERROR: this machine does not appear to have a password file" echo "" exit 1 fi

#get the appropriate password PASSWORD='cat $DBSE_HOME/.password.vertline.grep-i "${USER}".vertline. awk'{print $2}" BAD_SIDS ='cat $DBSE_HOME/.password.vertline.grep-i "${USER}".vertline. awd '{print $3}" OLD_PASS='cat $DBSE_HOME/.password.vertline.grep -i"${USER} ".vertline. awk '{print $4}" #if a password was found, print it to the screen if[$PASSWORD] then echo "${PASSWORD}" #print a message if passwords appear to not be synced between databases if[$BAD_SIDS] then BAD_SIDS ='echo ${BAD_SIDS}.vertline.cut -c2-100' echo "" echo "WARNING:\tThe password for ${USER} may not be synchronized" echo "\t\tbetween all databases. The database(s) ${BAD_SIDS} appear(s)" echo "t\tto use the old password '${OLD_PASS}'" echo "" fi else echo " " echo "ERROR: $ {USER} is not a supported username" echo " " exit 1 fi purge_records (SQL Script) SET ECHO ON SBT FEEDBACK ON SET FLUSH OFF SET HEADING OFF SET SERVEROUTPUT ON SET TERMOUT OFF SET VBRIFY OFF SPOOL /fiIes6/ipsa/purge/purge.sql.log REM *purge records in PERF_FACT_TB table delete from PERF_FACT_TB where PERF_FACT_TB.PERF_TIME_KEY_CD in (select PERF_METRIC_TIME_TB.PERF_TIME_KEY_CD from PERF_METRIC_TIME_TB where to_char(PERF_METRIC_TIME_TB.PERF_DT, 'MM/DD/YYYY') <=(select to_char(sysdate - 40, 'MM/DD/YYYY') from dual)); REM *Purge records in EVENTS_FACT_TB table delete from EVENTS_FACT_TB where EVENTS_FACT_TB.PERF_TIME_KEY CD in (select PERF_METRIC_TIME_TB.PERF_TIME_KEY_CD from PERF_METRIC_TIME_TB where to_char(PERF_METRIC_TIME_TB.PERF_DT, 'MM/DD/YYYY') <=(select to_char(sysdate -40, 'MM/DD/YYYY') from dual)); RBM *purge records in PERF_FACT_DLY_TB table delete from PERF_FACT_DLY_TB where PERF_FACT_DLY_TB.PERF_TIME_KEY_CD in (select PERF_METRIC_TIME_TB.PERF_TIME_KEY CD from PERF_METRIC_TIME_TB where to_char(PERF_METRIC_TIME_TB.PERF_DT, 'MM/DD/YYYY') <=(select to_char(sysdate - 397, 'MM/DD/YYYY') from dual)); SPOOL OFF

Detailed Description Paragraph Table (34):
#!/bin/sh #Script_name get_pass #------------------# #Description: This script will be used to retrieve the # #appropnate password from the .password # file. It can be used from the command # line to retrieve a password or from a # shell script to eliminate hard coding of # passwords. The .password file is located # #in $DBSE_HOME, with the executable # #located in $DBSE_HOME/bin.# # # #Dependencies: getpass requires one file, .password # # # #SID_FILE: Contains a list passwords for # #system, sys, and dbse.# # # #Command syntax: getpass USERNAME# #------------------# #what shell do we use? SHELL=/usr/bin/sh #where is our home? DBSE_HOME=/files0/ipsa/vendor/dbse/product/ 7.3.4; export DBSE_HOME #what path do we look for? PATH=/usr/bin: /bin: ${DBSE_HOME}/bin; export PATH #who are we retrieving the password for if[$1] then USER=$1 else echo echo "USAGE: getpass 'usemamep'" echo " " exit 1 fi #make sure the .password file exist if[!-s $DBSE_HOME/.password] then echo " " echo "ERROR: this machine does not appear to have a password file" echo" " exit 1 fi #get the appropriate password PASSWORD='cat $DBSE_HOME/.password.vertline.grep -i "${USER}".vertline.awk'{print $2}" BAD_SIDS='cat $DBSE_HOME/.password.vertline.grep -i "${USER}".vertline.awk '{print $3}" OLD_PASS='cat $DBSE_HOME/.password.vertline.grep -i "${USER}".vertline.awk '{print $4}" #if a password was found, print it to the screen if [$PASSWORD] then echo "${PASSWORD}" #print a message if passwords appear to not be synced between databases if[$BAD_SIDS] then BAD_SIDS='echo ${BAD_SIDS}.vertline.cut -c2-100' echo "" echo "WARNING:\tThe password for $ {USER} may not be sychronized" echo "\t\tbetween all databases. The database(s) $ {BAD_SID}appear(s)" echo "\t\tto use the old password '$ {OLD_PASS}'" echo "" fi else echo " " echo "ERROR: $ {USER}is not a supported usemame" echo " " exit 1 fi

Detailed Description Paragraph Table (35):
.backslash. Logs Log files generated from IPSA_SPSS.exe execution IPSA_VB_SPSS.log Error messages from program execution. .backslash. SPSS All SPSS related files. .backslash. Macros SPSS macros !incdef.SPS !bargrph.SPS bar graph !bxgrph.SPS boxplot !spect.SPS spectrum graph !xygrph.SPS xy line graph .backslash. Templates SPSS chart looks avail.clo availabilty graph bar.clo bar graph box.clo boxplot exception.clo exception graph line.clo xy line graph .backslash. Working Data Files SPSS macros use this directory for temporary files. .backslash. Web.backslash. HtmlTemplates HTML fragments for use to dynamically produce HTML todays_urls_head_template.txt batch TOC header todays_urls_detail.sub.-- batch TOC report title template.txt todays_urls_tail_template.txt batch TOC footer detail_urls_head_template.txt batch/adhoc list-of-reports header detail_urls_detail_template.txt batch/adhoc list-of-reports details detail_urls_tail_template.txt batch/adhoc list-of-reports footer <SAWEB> The report output home directory, as specified in <SARPHME>.backslash. InputQueue.backslash. IPSA_Reporting.sub.-- Config.txt (see Configuration). .backslash. Adhoc.backslash. <date>_INFO Adhoc reports issued on the date <date> where <date> is of the form YYYYMMDD. This directory is created by the report generation script. <adhoc_report_title><date><unique id>.html the generated adhoc report .backslash. Batch.backslash. Daily Daily batch reports home directory .backslash. <date>_INFO

Generated batch reports. <date> is of the form YYYYMMDD, but is the date prior to the date when the report request was issued. This directory is created by the report generation script. .backslash. Images Images for the HTML pages Graph1.gif list bullet SA1.gif Service Assurance logo <WEB ROOT> Root directory of the web server where the reports will be housed. bground.gif background image for the adhoc web pages .backslash. cgi-bin Adhoc files location. This directory must be accessible from a web browser via HTTP and have "execute" permission assigned to it. adhoc.pl Perl script used to generate ad hoc reports. This file is accessed via HTTP from a web browser to start the ad hoc report data gathering process. adhoc.setup.txt Setup and configuration file for the adhoc.pl script. Contains report types and descriptions.

Detailed Description Paragraph Table (42):
TABLE 16 Low Level Daily Event Exceptions REQUIRED FILES FOR REPORTING IPSA_StartSPSS.cmd Located anywhere, it is used to start the reporting process (see Configuration). Instantiates two SPSS processes (spssw.exe, spsswin.exe) and IPSA_SPPS.exe <SABATCH> Batch queue file location, as specified in the IPSA_StartSPSS.cmd file (see Configuration). IPSA_BatchQueue.txt Queue file that, if present, is passed to SPSS and contains batch report specifications. This file is deleted after being read by IPSA_SPSS.exe. See Adhoc.doc and graphs.doc for content explanations. <SAADHOC> Adhoc queue file location, as specified in <SARPHME>.backslash.InputQueue.backslash.IPSA_Reporting_Config.txt (see Configuration). IPSA_AdhocQueue.txt Queue file passed to SPSS that contains ad hoc report parameters. This file is created by the web interface (<WWW ROOT>.backslash. cgi-bin.backslash. adhoc.pl) and deleted after use by IPSA_SPSS.exe. See Adhoc.doc and graphs.doc for content explanations. Adhoc working file with a unique IPSA_SPSS.exe IPSA_AdHocQueue_Working.sub.-- instance number, <instance>, that is defined in <instance>.txt IPSA_StartSPSS.cmd (see Configuration). The presence of this file notifies IPSA_SPSS.exe to IPSA_ProcessTheBatchQueue.txt read <SABATCH>.backslash. IPSA_BatchQueue.txt and pass it to SPSS. This file is polled for every five seconds. <SARPHME> The reporting home directory, as specified in the IPSA_StartSPSS.cmd file (see Configuration). All files needed to generate reports are located within this directory. .backslash. InputQueue IPSA_SPSS.exe configuration and control files. IPSA_Reporting_Config.txt IPSA_SPSS.exe configuration file. Includes database location and login information, output directories and image format specification for the generated graphs (see Configuration). IPSA_ControlQueue.txt Contains commands to be issued to IPSA_SPSS.exe during execution. Used for terminating the process. .backslash.Logs Log files generated from IPSA_SPSS.exe execution IPSA_VB_SPSS.log Error messages from program execution. .backslash. SPSS All SPSS related files. .backslash. Macros SPSS macros !incdef.SPS !bargrph.SPS bar graph !bxgrph.SPS boxplot !spect.SPS spectrum graph !xygrph.SPS xy line graph .backslash. Templates SPSS chart looks avail.clo availabilty graph bar.clo bar graph box.clo boxplot exception.clo exception graph line.clo xy line graph .backslash. Working Data Files SPSS macros use this directory for temporary files. .backslash. Web.backslash. HtmlTemplates HTML fragments for use to dynamically produce HTML batch TOC header todays_urls_head_template.txt batch TOC report title todays_urls_detail_template.txt batch TOC footer todays_urls_tail_template.txt batch/adhoc list-of-reports header detail_urls_head_template.txt batch/adhoc list-of-reports details detail_urls_detail_ template.txt detail_urls_tail_template.txt batch/adhoc list-of-reports footer <SAWEB> The report output home directory, as specified in <SARPHME>.backslash.InputQueue.backslash.IPSA_Reporting_Config.txt (see Configuration). .backslash. Adhoc.backslash. <date>_INFO Adhoc reports issued on the date <date> where <date> is of the form YYYYMMDD. This directory is created by the report generation script. the generated adhoc report <adhoc_report_title><date><unique id>.html .backslash. Batch.backslash. Daily Daily batch reports home directory .backslash.<date>_INFO Generated batch reports. <date> is of the form YYYYMMDD, but is the date prior to the date when the report request was issued. This directory is created by the report generation script. .backslash. Images Images for the HTML pages Graph1.gif list bullet SA1.gif Service Assurance logo <WEB ROOT> Root directory of the web server where the reports will be housed. bground.gif background image for the adhoc web pages .backslash. cgi-bin Adhoc files location. This directory must be accessible from a web browser via HTTP and have "execute" permission assigned to it. adhoc.pl Perl script used to generate ad hoc reports. This file is accessed via HTTP from a web browser to start the ad hoc report data gathering process. adhoc.setup.txt Setup and configuration file for the adhoc.pl script. Contains report types and descriptions.

Detailed Description Paragraph Table (43):
.backslash. Logs Log files generated from IPSA_SPSS.exe execution IPSA_VB_SPSS.log Error messages from program execution. .backslash. SPSS All SPSS related files. .backslash. Macros SPSS macros !incdef.SPS !bargrph.SPS bar graph !bxgrph.SPS boxplot

!spect.SPS spectrum graph !xygrph.SPS xy line graph .backslash. Templates SPSS chart looks avail.clo availabilty graph bar.clo bar graph box.cloboxplot exception.clo exception graph line.clo xy line graph .backslash.Working Data Files SPSS macros use this directory for temporary files. .backslash. Web.backslash. HtmlTemplates HTML fragments for use to dynamically produce HTML todays_urls_head_template.txt batch TOC header todays_urls_detail_template.txt batch TOC report title todays_urls_tail_template.txt batch TOC footer detail_urls_head_template.txt batch/adhoc list-of-reports header detail_urls_detail_template.txt batch/adhoc list-of-reports details detail_urls_tail_template.txt batch/adhoc list-of-reports footer <SAWEB> The report output home directory, as specified in <SARPHME>.backslash. InputQueue.backslash. IPSA_ Reporting_ Config.txt (see Configuration). .backslash. Adhoc.backslash. <date>_INFO Adhoc reports issued on the date <date> where <date> is of the form YYYYMMDD. This directory is created by the report generation script. <adhoc_report_title><date><unique id>.html the generated adhoc report .backslash. Batch.backslash. Daily Daily batch reports home directory .backslash.<date>_INFO Generated batch reports. <date> is of the form YYYYMMDD, but is the date prior to the date when the report request was issued. This directory is created by the report generation script. .backslash.Images Images for the HTML pages Graph1.gif list bullet SA1.gif Service Assurance logo <WEB ROOT> Root directory of the web server where the reports will be housed. bground.gif background image for the adhoc web pages .backslash.cgi-bin Adhoc files location. This directory must be accessible from a web browser via HTTP and have "execute" permission assigned to it. adhoc.pl Perl script used to generate ad hoc reports. This file is accessed via HTTP from a web browser to start the ad hoc report data gathering process. adhoc.setup.txt Setup and configuration file for the adhoc.pl script. Contains report types and descriptions.

Detailed Description Paragraph Table (64):
TABLE 19 Identify and Describe Requirement Description: Provide information regarding the require- ments, plan and implementation of Incident Reporting on the SA project Scenarios: Type: Business Process Flow: Overall Rating: Business Need Desc: Affected Parties: Service Assurance Team Project Sponsor: Network Line of Business Existing/New: External Dependencies: Method for Verification: In Scope: Service Assurance Internal Initiative (Phase 2)

Detailed Description Paragraph Table (72):
Configuration OfVariables # # Instantiate Local Variables # - Make the variables ("my") local to keep them out of # the global name space. Single step alarm, so no # need for global variables. # my $EpochTime=0; my @DaysPerMonth = (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31); my $year,$mon,$day,$hour,Smin,$sec; my $TZAdjust,$DSTAdjust; my $ShowDebug,$NumLeapYears,$SecondsThisInterval; my $SnmpTrapTimestamp,$SnmpTrapNode, $SA_CorrEnt_NotifyOvTrapString ; # Initialize Date/Time Variables # - 2d line turns year into 4-digit year (i.e. 19xx) # as localtime function returns year-1900. Timestamp # design is standard for phase 2 development. # $year = ((localtime( ))[5]); $mon = ((localtime( ))[4]); $day = ((localtime( ))[3]); $hour = ((localtime( ))[2]); $min = ((localtime( ))[1]); $sec = ((localtime( ))[0]); # Determine number of seconds for years if($year < 200) { # year is in two digit form if($year < 90) { # year is presumably in the 21st century $year += 2000; } else { # year is presumably in the 20th century $year += 1900; } } # Do not return negatives - and if2100 or later, add to leap year calcs if($year < 1970 .vertline..vertline. $year >= 2100) { return 0; } $year -= 1970; $SecondsThisInterval = $year * 365 * 24 * 60 * 60; # leap year calculations $NumLeapYears = int(($year+2)/4); if(($year+2) % 4 == 0) { # this is a leap year (assuming 1970-2099) # check if we have hit feb 29 yet: # recall, mon is 0-based if($mon <= 1) { # we need to subtract a year, as this is jan $NumLeapYears--; } } $SecondsThisInterval += $NumLeapYears * 24 * 60 * 60; $EpochTime += $SecondsThisInterval; # Determine number of seconds for months so far while($mon--) { $EpochTime += $DaysPerMonth[$mon] * 24 * 60 * 60; } # Determine number of seconds for days so far $EpochTime += ($day - 1) * 24 * 60 * 60; # Determine number of seconds for hours so far $EpochTime += $hour * 60 * 60; # Determine number of seconds for minutes so far $EpochTime += $min * 60; # Determine number of seconds for seconds so far $EpochTime += $sec; # Adjust for time zone (which should be of the form-12:00 .. 12:00) # THIS SHOULD USE ACTUAL TIME ZONE CODES $TZAdjust = "5:00"; if($TZAdjust =.about./ (-?)(.backslash.d):(.backslash.d*)$/) { # Time Zone adjustment needs to subtract if($1) { $EpochTime -= ($2 * 60 * 60) + ($3 * 60); } else { $EpochTime += ($2 * 60 * 60) + ($3 * 60); } } # Adjust for day light savings time # Fire The Trap Into HP OpenView # # The 6th value on the trapgen string (right next to # the generic trap number (6)) is the specific trap # number. This has to be associated with some event # configured in HP OpenView. (i.e. 6 3 105 - The `3` # is the specific trap ID) # system("/opt/seasoft/bin/trapgen nsmmws16 1.3.6.1.4.1.78 $A 6 4 105 1.3.6.1.4.1.78.0.4 octetstringascii 1000003 1.3.6.1.4.1.78.0.4 octetstringascii $EpochTime

1.3.6.1.4.1.78.0.4 octetstringascii 1"); # # Subroutine: SA_ifEntry_C2Rate_Util # Overview: Counter To Rate Conversion Script # Builds Utilization Percentage And # Rate For Interface Instances. # # # # Configuration # # Declare Local Variables # my $ifSpeed,$Old_ifInOctets,$Old_ifOutOctets; my $New_ifInOctets,$New_ifOutOctets; my $Old_ifInOctets,$Old_ifOutOctets; my $Delta_ifInOctets,$Delta_ifOutOctets; my $Summed_Octets,$Converted_To_Bits; my $New_Timestamp,$Old_Timestamp,$Delta_Time; my $BPS,$Utilization_Percentage; my $sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst; my $Maximum_Counter_Size = 4294967295; # Construct Initial Time Variables ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time); # Fix 19xx Year $year = 1900 + $year; $New_Timestamp = "$year-$mon-$mday $hour:$min:$sec"; $Old_Timestamp = $Saved Timestamp{$pollkey}; # Get New Interface Values $ifSpeed = ifEntry.ifSpeed; $New_ifInOctets = ifEntry.ifInOctets; $New_ifOutOctets = ifEntry.ifOutOctets; # Retrieve Old Interface Values $Old_ifInOctets = $Saved_ifInOctets{$pollkey}; $Old_ifOutOctets = $Saved_ifOutOctets{$pollkey}; # # # Data Manipulation # # Gather And Build ifInOctets Delta # $Delta_ifInOctets = $New_ifInOctets - $Old_ifInOctets; if($Delta_ifInOctets < 0){ $Delta_ifInOctets = ($Maximum_Counter_Size - $Old_ifInOctets) + $New_ifInOctets; } # Gather And Build ifOutOctets Delta # $Delta_ifOutOctets = $New_ifOutOctets - $Old_ifOutOctets; if($Delta_ifOutOctets<0) { $Delta_ifOutOctets = ($Maximum_Counter_Size - $Old_ifOutOctets) + $New_ifOutOctets; } # Gather And Build Timestamp Delta # - Octets Are Made Of 8 Bits - i.e. The Conversion # $Summed_Octets = $Delta_ifInOctets + $Delta_ifOutOctets; $Converted_To_Bits = $Summed_Octets * 8; $Delta_Time = SubtractTime($Old_Timestamp,$New_Timestamp); # Do The Math - Create Deliverable Values (Rate And Counter) # $BPS = $Converted_To_Bits/$Delta_Time; $Utilization_Percentage = ($BPS/$ifSpeed) * 100; # # # Report Data Findings To Log File # # Open The Log File And Write The Data To Disk # # Open(LOG,">>/opt/seasoft/userfiles/logs/SA_ifEntry_C2Rate_ Util.dat"); if(LOG) { printf LOG "%s,InterfaceUtilization_Percent,%s,%s,%.3f.backslash.n",$N,$OI, $New_Timestamp, $Utilization_Percentage; printf LOG "%s,InterfaceUtilization_BPS,%s,%s,%.3f.backslash.n",$N,$OI, $New_Timestamp,$B PS; close LOG; } else { # # If An Error On Opening The Log File Then Open Error File # And Write To Disk # open(LOG,">>/usr/seasoft/userfiles/logs/SA_if_Entry_C2Rate_ Util.err"); print LOG "$New_Timestamp:Unable To Write To Log SA_ifEntry_C2Rate_Util.log"; close LOG; } # # # Update The Global Data Hashes # $Saved_ifInOctets{$pollkey} = $New_ifInOctets; $Saved_ifOutOctets {$pollkey} = $New_ifOutOctets; $Saved_Timestamp {$pollkey} = $New_Timestamp; # # # # # Subroutine - Builds Deltas For Use By Main Handler # Script. Inputs timestamps in form of # `YYYY-MM-DD hh:mm:ss` and returns # delta in form of seconds elapsed. # sub SubtractTime { # Declare Local Variables my $date1,$date2,$year1,$year2,$months1; my $months2; my $days1,$days2,$time1,$time2,$hours1; my $hours2,$mins1,$mins2,$secs1,$secs2; my $result1,$result2,$DifferenceInSeconds; # assuming YYYY-MM-DD hh:mm:ss format # (24hr representation) # Convert Date/Time To Seconds ($date1,$time1) = split(" ",$_[0]); ($year1,$months1,$days1) = split("-",$date1); ($hours1,$mins1,$secs1) = split(":",$time1); $result1 = $year1*31104000+$months1 *2592000+$days1*86400+$hours1*3600+ $mins1*60+$secs1; # assuming YYYY-MM-DD hh:mm:ss format # (24hr representation) # Convert Date/Time To Seconds ($date2,$time2) = split(" ",$_[1]); ($year2,$months2,$days2) = split("-",$date2); ($hours2,$mins2,$secs2) = split(":",$time2); $result2 = $year2*31104000+$months2*2592000+$days2*86400+$hours2*3600+ $mins2*60+$secs2; # Get Delta $DifferenceInSeconds = $result2 - $result1; return($DifferenceInSeconds); } # # Subroutine: SA_NodeMonitor_BuildTrap # Overview: Trap VarBind Builder # # # # Configuration # # Delcare Local Variables # my $sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst; ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time); my $Timestamp,$Node; # Assign Values To Variables # $Node = $A; $year = 1900 + $year; $Timestamp = "$year-$mon-$mday $hour:$min:$sec"; # Build The Trap VarBind

Detailed Description Paragraph Table (104):
TABLE KK SACommon variable definition example #These variable is used with the data retrieval script $NodeList = "/sa/dev/glueware/bin/node_list.def"; $DestDir = "/sa/dat"; $DestHost = "twmmdb02"; $CopyCommand = "rcp -p"; Table LL (the bold commands are examples used with SACommon.pm) use SACommon; open (NODE_F, $SACommon::NodeList) .parallel. die "error could not open $SACommon::NodeList";

Detailed Description Paragraph Table (109):
TABLE 36 Identify and Describe Requirement Description: Hardware inventory, Software install locations, and memory/disk-space reqs Scenarios: Type: Business Process Flow: Overall Rating: Business Need Desc: Affected Parties: Service Assurance Team Project Sponsor: Network Line of Business Existing/New: External Dependencies: Method for

Verification: In Scope:

Detailed Description Paragraph Table (112):
TABLE 39 Service Assurance Application Memory and System Requirements Suggested Memory Program Requirements Suggested System Requirements Collector RAM: Hardware Requirements: Object Server: 50 SUN Workstations: will run all components of Collector. Mbytes per server SPARC 20 or better, Ultra 1 or better, with appropriate (dependant upon RAM and disk. (recommended system for desktop only: number of events SPARC 5 or better. in the Object HP Workstations: will run all components of Collector. server) C110 workstation or better, D230, K210, T520 servers or Desktops: better, with appropriate RAM and disk. 10 Mbytes per AIX Workstations: AIX .upsilon.3.2 will run Generic, NetView desktop (standard and Syslog probes, with appropriate RAM and disk. AIX user), 15 Mbytes 4.1.2 and above will run all 3.2.1 components. per desktop NT Workstations: NT probes and EventList at present, (standard user plus with appropriate RAM and disk. objective view), 20 Disk Space: Mbytes per The following table lists the amount of disk storage required to desktop (above store components of the Collector system. In some cases, plus administrator further economies may be possible, i.e., by discarding unused tools) probes. Probes: 5 Mbytes JEL per probe Common Object Process Gate Probes daem Gateways: 15 Platform/OS Files Desktop Server Control way (all) on Mbytes per SunOS 4.1.x 0.7 M 30 M 1.8 M 1 M 1.5 M 19 M 7 M gateway Sun Solaris 0.9 M 12 M 2.5 M 1.4 M 2.0 M 23 M 7 M Java Event List: 5 2.x Mbytes per HP-UX 9.07 0.8 M 11 M 1.9 M 1 M 1.6 M 18 M 7 M daemon HP-UX 0.8 M 11 M 2.0 M 1 M 1.6 M 16 M 7 M Web server: 2.5 10.10 and Mbytes per user 10.20 typical web AIX 0.8 M -- -- 1.5 M -- 4 M -- server, will vary) Windows -- -- -- -- -- 6 M -- NT In addition 10-20 Mb should be allowed for logging space on systems running Gateways, Object Servers, Probes or Process Control. Reporter It is recommended that you run the Collector server with a minimum of 128 MB and a maximum of 256 MB of memory. These values recommendations but if you have the potential for large amounts of data to report on, this is going to be vital to the efficiency of the application. This section describes what action to take on the server if you do not have these values set and you are getting memory errors while running Collector/ Reporter. See page 32 of the Admin/user guide of Reporter for making adjustments to memory for UNIX. PATROLL The CPU and Operating System Requirements: ERLER PATROLLERLER Sun SPARC; Solaris2; Min version 2.4; Solaris2-sun4 Console should be run Sun SPARC; Solaris2; Min version 2.5; Solaris25-sun4 on a machine with at nls is a system prerequisite for Sun O/S 4.xx installations least 64 MB of Disk Space: memory. Each PATROLLERLER Console requires about 20 MB of disk space. The Console also requires an additional 31 MB of disk space for the supporting files such as icon images and online help files. You will need an additional 27 MB of disk space if you choose to install the optional background images for European country maps. Each PATROLLERLER Agent requires about 10 MB of disk space. Each PATROLLERLER Event Manager (PEM) Console requires about 5 MB of disk space. If the PEM Console is installed independently of the PATROLLER Console, then an additional 24 MB of disk space is required for the supporting files such as icon images and online help files. PATROLLERLER Module space requirements vary. The installation script furnishes an estimate of each module's requirements: Event ii. Hardware Configuration: (minimum UNIX) Correlator 48 MB for the 200 MB disc space color monitor 1024 .times. 768 and server Solaris 2.5.1 or HP/UX 10.20 Manager 32 MB for the Hardware Configuration: (minimum NT) client P5-166 Intel Processor, 40 MB disc space, color monitor NT 1024 .times. 768 32 MB RAM Microsoft Windows NT 4.0 Note: Supports the following: OpenView Network Node Manager - versions 4.11 and 5.01 OpenView IT/Operations - version 4.0 for HP-UX HP iii. Unix OpenView 64 Mbytes Computer: Unix recommended Use one of the following computers as the NNM Version minimum Management Station. 5.01 32 Mbytes HP 9000 Series 700 NT minimum for HP 9000 Series 800, J and K models Version NNM 250 Sun SPARCstation 5,10,20,2000 5.02 Note: The amount of Sun SPARCclassics RAM in your NNM Sun Servers management station Graphics Dipslay should be based on the X Terminal or Workstation graphics display with number of nodes 1280 .times. 1024 resolution, 8 color planes (recommended) which you wish to 1024 .times. 768 resolution, 6 color planes (minimum) manage. Additional 20" display RAM may also be Installation Device required to run third- CD-ROM drive party OpenView Disk Space applications on top of The minimal disk space for NNM installation is shown NNM. See the below Network Node HP-UX 9.x - 85 Mbytes Manager Performance HP-UX 10.x - 85 Mbytes and Configuration Solaris 2.x - 130 Mbytes Guide for assistance in Operating System calculating for the One of the operating systems listed below must be running optimum amount of on the NNM mgmt system. RAM. HP-UX 9.0-9.07 (9.x) iv. HP-UX 10.01, 10.10, and 10.20 (10.x) 32 Mbytes of Solaris 2.4 and 2.5.x RAM to manage Networking Subsystem 250 nodes and, The appropriate TCP/IP networking subsystems (e.g. LAN 48 Mbytes of Link, ARPA Services) found within the operating system RAM to manage must be installed and configured to yield TCP/IP network up to 2500 nodes. connectivity Note: You will need connectivity. to have a minimum Windowing Subsystem amount of paging file HP-UX: X Windows/Motif size (available virtual Solaris: OpenWindows memory) configured.

SNMP Agent If NNM is being The NNM management station must be running an SNMP installed as a remote agent. An SNMP agent is shipped with NNM for HP-UX 9.x console, Paging Files and Solaris systems, and is automatically installed when is checked to be at installing NNM. HP-UX 10.x systems use the SNMP agent least 50 Mbytes. If shipped with the operating system. this is not a remote NT console installation, Operating System 60 Mbytes will be the You should be running Windows NT 3.51 or Windows NT minimum. 4.0 for NNM or higher to run successfully Graphics Display Your screen resolution must be at least 600 .times. 800 to support NNM display objects. Networking Subsystem You should have TCP/IP services installed. Platinum v. VCI (important component of CCC/Harvest) Technology Server Microsoft Windows 95 or Windows NT CCC/Harvest 16 Mb main Tool that supports Microsoft's Common Source Code memory Control (SCC) Interface. Following is a partial list of It is recommended SCC-compliant tools: that 2 Mb of Visual C++ 4.2 and 5.0 virtual memory is Visual Basic 4.0 and 5.0 allocated for each Visual J++ 1.1 user Paradigm Plus 3.5.1 Client (Solaris) PowerBuilder 5.0.03 SPARCstation or Unix SPARCserver CD-ROM drive, 8 mm tape drive, 4 mm DDS cartridge, or running Solaris 2.5 1/4 inch cartridge tape drive (SunOS 5.5) or Oracle RDBMS version 7.3 or beyond, including the with X-Windows following options: System Version SQL*Plus, PL/SQL, SQL*Loader, Pro-C, SQL*Net 11R5. Note: HP-UX 10 requires Oracle 7.3.4 or beyond. Approximately 50 NT MB of disk space IBM-compatible computer with a 486, or Pentium is required for the processor installation process Network connection to a Unix or Windows NT-based of the server using the TCP/IP protocol CCC/Harvest product files. vi. Server At least 12 Mb of free hard drive space A minimum of 32 Mb main memory. The Oracle database and CC/Harvest Broker together require about 14

Detailed Description Paragraph Table (116):
Building the Service Assurance Environment Network Management Station System Build Verify system requirements Create `noc` user Create `netman` group Install HP OpenView Network Node Manager (NNM) Install Patroller Patroller Product Licensing Install Patroller Console Install PatrolView Install Patroller Agents Install ECM Install SAS* Telalert Installation & Configuration* Install Netscape FastTrack Server* Install Oracle* Install Perl* Install Perl Modules* Install custom scripts Event Handling System (EHS) setup and configuration EHS component test Network Availability (NA) setup and configuration NA component test Process Availability (PA) setup and configuration PA component test Service Availability (SA) setup and configuration SA component test Reporting System (RS) setup and configuration RS component test (Reporting Test Cases.doc)

CLAIMS:

1. A method for providing service assurance for a network to maintain a predetermined agreed upon Quality of Service, comprising the steps of: (a) generating an alarm to indicate a status of a network; wherein the step of generating an alarm to indicate a status of a network further comprises the steps of: selecting a parameter of the network that is to be monitored, determining a triggering level of the parameter, monitoring the parameter of an occurrence of the triggering level, and initiating an alarm notification upon the monitored occurrence of the triggering level; (b) dispatching network event information of the network upon generation of the alarm; (c) mapping the network event information; (d) manipulating data collected on the status of the network, wherein manipulating data comprises: (i) concatenating data collected on a network into a master file; (ii) reformatting the concatenated data into a standardized format; (iii) translating the standardized data to key codes; (iv) sorting the translated data according to predetermined criteria; and (v) concatenating the sorted data together; (e) storing the manipulated data in a database; and (f) graphically conveying availability of the network.

2. A method as recited in claim 1, wherein the step of dispatching network event information of the network upon generation of the alarm further comprises the steps of: monitoring a network for an event; generating at least one notification action based upon the occurrence of the event, wherein the notification action comprises at least one of: an alphanumeric page, an e-mail message, a resolution script, a remedy trouble ticket, and a log message; and transmitting the notification action to notify a recipient about the occurrence of the event.

3. A method as recited in claim 1, wherein the step of mapping the network event information further comprises the steps of: monitoring a network for the occurrence of availability events, threshold events, and trap events, correlating at least one occurred event to at least one other occurred event to generate at least one correlating event, mapping the occurred events and correlating events on at least one network map; and displaying the network map.

5. A computer program embodied on a computer readable medium for providing service assurance for a network to maintain a predetermined agreed upon Quality of Service, comprising: (a) a code segment for generating an alarm to indicate a status of a network; wherein the code segment for generating an alarm to indicate a status of a network is further adapted for selecting a parameter of the network that is to be monitored, determining a triggering level of the parameter, monitoring the parameter of an occurrence of the triggering level, and initiating an alarm notification upon the monitored occurrence of the triggering level; (b) a code segment for dispatching network event information of the network upon generation of the alarm; (c) a code segment for mapping the network event information; (d) a code segment for manipulating data collected on the status of the network, wherein manipulating data comprises: (i) concatenating data collected on a network into a master file; (ii) reformatting the concatenated data into a standardized format; (iii) translating the standardized data to key codes; (iv) sorting the translated data according to predetermined criteria; and (v) concatenating the sorted data together; (e) a code segment for storing the manipulated data in a database; and (f) a code segment for graphically conveying availability of the network.

6. A computer program as recited in claim 5, wherein the code segment for dispatching network event information of the network upon generation of the alarm is further adapted for monitoring a network for an event; generating at least one notification action based upon the occurrence of the event, wherein the notification action comprises at least one of: an alphanumeric page, an e-mail message, a resolution script, a remedy trouble ticket, and a log message; and transmitting the notification action to notify a recipient about the occurrence of the event.

7. A computer program as recited in claim 5, wherein the code segment for mapping the network event information is further adapted for monitoring a network for the occurrence of availability events, threshold events, and trap events, correlating at least one occurred event to at least one other occurred event to generate at least one correlating event, mapping the occurred events and correlating events on at least one network map; and displaying the network map.

9. A system for providing service assurance for a network to maintain a predetermined agreed upon Quality of Service, comprising: (a) logic for generating an alarm to indicate a status of a network; wherein the logic for generating an alarm to indicate a status of a network is further adapted for selecting a parameter of the network that is to be monitored, determining a triggering level of the parameter, monitoring the parameter of an occurrence of the triggering level, and initiating an alarm notification upon the monitored occurrence of the triggering level; (b) logic for dispatching network event information of the network upon generation of the alarm; (c) logic for mapping the network event information; (d) logic for manipulating data collected on the status of the network, wherein manipulating data comprises: (i) concatenating data collected on a network into a master file; (ii) reformatting the concatenated data into a standardized format; (iii) translating the standardized data to key codes; (iv) sorting the translated data according to predetermined criteria; and (v) concatenating the sorted data together; (e) logic for storing the manipulated data in a database; and (f) logic for graphically conveying availability of the network.

10. A system as recited in claim 9, wherein the logic for dispatching network event information of the network upon generation of the alarm is further adapted for monitoring a network for an event; generating at least one notification action based upon the occurrence of the event, wherein the notification action comprises at least one of: an alphanumeric page, an e-mail message, a resolution script, a remedy trouble ticket, and a log message; and transmitting the notification action to notify a recipient about the occurrence of the event.

11. A system as recited in claim 9, wherein the logic for mapping the network event information is further adapted for monitoring a network for the occurrence of availability events, threshold events, and trap events, correlating at least one occurred event to at least one other occurred event to generate at least one correlating event, mapping the occurred events and correlating events on at least one network map; and displaying the network map.